toggle toc
show orig
show new
show both

Only reserve names of optional functions if necessary

Jens Gustedt, INRIA, France

org:	ISO/IEC JCT1/SC22/WG14	document:	N2839
target:	IS 9899:2023	version:	1
date:	2021-10-11	license:	CC BY

1. Introduction

C23 in its current form adds a large number of functions (in the order of some hundred) for decimal floating point types (DFP) or binary floating points (BFP) without following any naming scheme that the standard established. This is particularly disturbing, because as of now external names are reserved regardles if the corresponding header is included.

By that, currently valid programs for C17 may become invalid even if they don't use the decimal or binary floating point options. Even worse, because this implies linking (and thus for example link order) violations of the use of the names of these functions could go undetected for a long time and even result in programs with undefined behavior.

Generally, we have now several sets of external identifiers that are optional, but for which the names are nevertheless reserved. In view of the recent progress for C23 to distinguish potentially reserved identifiers from that set, it seems in order relax our constraints when either the implementation does not implement the option, or when the program doesn't use it at all. We identified the following sets of external identifiers that are optional:

- All the functions with complex parameters, <*complex.h*>
- All the functions with decimal floating point arguments, <*fenv.h*>,<*math.h*> and <*stdlib.h*>
- All the functions of the atomics option, <*stdatomic.h*>
- All the functions of the threads option, <*threads.h*>
- All the functions using the binary floating point optional types, Annex F

Similar to the additions in Annex K, we propose to break this monolithicity of the C library, and to only reserve those identifiers if a programs actively refers to at least one of these sets. We only explicitly propose wording for complex, DFP and BFP functions, in their main headers since these follow no naming scheme of the standard. Optionally WG14 could also take the same approach for <*fenv.h*>,<*math.h*>,<*stdlib.h*>, <*stdatomic.h*> and <*threads.h*>, but this would require a new paper.

2. Changes

2.1. Change in 7.3 (Complex arithmetic <complex.h>) p2

2 Implementations that define the macro `__STDC_NO_COMPLEX__` *need not provide this header nor support any of its facilities. All identifiers with external linkage in any of the following subclauses are reserved for use as identifiers with external linkage if any of them is used by the program (either directly or via a type-generic macro in <tgmath.h>) to refer to the function as described in this clause. None of them is reserved if* `__STDC_NO_COMPLEX__` *is defined or if the program uses none of them to refer to the functions described in this clause.*[FNT0)]

FNT0) See Annex B.2 for a list of declarations of these identifiers.

2.2. Change in 7.12 (Mathematics <math.h>) p2

2 The feature test macro `__STDC_VERSION_MATH_H__` *expands to the token yyyymmL. If the implementation defines* `__STDC_IEC_60559_DFP__` *all identifiers with external linkage in any of the following subclauses that use the decimal floating point types in their declaration are reserved for use as identifiers with external linkage if any of them is used by the program (either directly or via a type-generic macro in <tgmath.h>) to refer to the function as described in this clause. None of them is reserved if* `__STDC_IEC_60559_DFP__` *is not defined or if the program uses none of them to refer to the functions described in this clause.*[FNT1)]

FNT1) See Annex B.11 for a list of declarations of these identifiers.

2.3. Change in 7.25 (Type-generic math <tgmath.h>) p2

3 This clause specifies a many-to-one correspondence of functions in <math.h> and <complex.h> with type-generic macros.[338)] *Use of a type-generic macro invokes a corresponding function whose type is determined by the types of the arguments for particular parameters called the generic parameters.*[339)] *An implicit use of such a function through the corresponding type-generic macro constitutes a usage of the external name as if explicit.*

2.4. Change in F.1 (IEC 60559 floating-point arithmetic, Introduction) p7

This annex amends some standard headers with declarations or definitions of identifiers contingent on whether certain macros whose names begin with `__STDC_WANT_IEC_60559_` *and end with* `_EXT__` *are defined (by the user) at the point in the code where the header is first included. Within a preprocessing translation unit, the same set of such macros shall be defined for the first inclusion of all such headers. All identifiers with external linkage in any of the following subclauses that are added due to the definition of one of these macros are reserved for use as identifiers with external linkage if any of them is used by the program to refer to the function as described in this annex. None of them is reserved if the macro* `__STDC_IEC_60559_BFP__` *is not defined or if the program uses none of them to refer to the functions described in this annex.*[FNT2)]

FNT2) See Annex B.11 for a list of declarations of these identifiers.

2.5. Impact

These changes do not invalidate user code. They maintain validity of user code written for C17 and that does not use decimal or binary floating point options, even if they coincidentaly define identifiers of the incriminated sets as external symbols.

Implemenations that start to offer the decimal floating point option or the binary floating point option of Annex F, have to ensure that they protect user code that does not use these options from naming clashes for these symbols. This can be achieved for example by shipping these options as separate library objects or by definining the symbols

as “weak”.

3. Questions for WG14

1. Shall we integrate Change 2.1 into C23?
2. Shall we integrate Change 2.2 into C23?
3. Shall we integrate Change 2.3 into C23?
4. Shall we integrate Change 2.4 into C23?
5. Does WG14 want exempt the reservation of the optional functions of <*fenv.h*>,<*math.h*>,<*stdlib.h*>, <*stdatomic.h*> and <*threads.h*>, in a similar way?

4. Note to the editors

The formulation for the `WANT` macros in Annex F is quite strange. It refers to several such macros where there actually is only one. If Change 2.2 is approved by WG14, it might be a good opportunity to clean that up. The change could then be

> *This annex amends some standard headers with declarations or definitions of identifiers contingent on whether ~~certain macros whose names begin with __STDC_WANT_IEC_60559_ and end with _EXT__ are~~ the macro __STDC_WANT_IEC_60559_**EXT**__ is defined (by the user) at the point in the code where the header is first included. Within a preprocessing translation unit, the ~~same set of such macros~~ macro shall be defined for the first inclusion of all such headers or not at all. All identifiers with external linkage in any of the following subclauses that are added due to the definition this macro are reserved for use as identifiers with external linkage if any of them is used by the program to refer to the function as described in this annex. None of them is reserved if the macro __STDC_IEC_60559_**BFP**__ is not defined or if the program uses none of them to refer to the functions described in this annex.*[FNT2)]
>
> *[FNT2)] See Annex B.11 for a list of declarations of these identifiers.*